Data Munging

Project Description

Your team has been tasked with collecting metrics on a plethora of disparate shipping data. This task comes straight from the top, so it would be wise to give it your all. The data is contained in a number of different spreadsheets, each with its own competing schema. In order to interrogate the data, all of it has to be in the same place and in the same format. Currently, the shipping data exists in several places in several formats and is therefore impossible to query. To combine the spreadsheets, you need to write a python script to read through every row, extract the pertinent data, figure out how to combine it, munge it into the right format, and upload it to the database. Plenty of steps, but the resulting data will be much easier to query. Once the database contains all the data, you can pass it off to the analysis team to extract all the relevant metrics. Good luck!

Data Dictionary

Part 1: Get the data

First, you need to get your hands on the relevant data. The shipping department has been kind enough to provide you with a repository containing all of their spreadsheets, as well as a copy of the sqlite database. First, fork and clone the repository at: https://github.com/theforage/forage-walmart-task-4

Part 2: Populate the database

Your task is to insert all of the data contained in the provided spreadsheets into the SQLite database. You will write a Python script which:

- Reads each row from the spreadsheets.
- Extracts the relevant data.
- Munges it into a format that fits the database schema.
- Inserts the data into the database.

Spreadsheet 0 is self contained and can simply be inserted into the database, but spreadsheets 1 and 2 are dependent on one another. Spreadsheet 1 contains a single product per row, you will need to combine each row based on its shipping identifier, determine the quantity of goods in the shipment, and add a new row to the database for each product in the shipment. The origin and destination for each shipment in spreadsheet 1 are contained in spreadsheet 2. You may assume that all the given data is valid - product names are always spelled the same way, quantities are positive, etc. When you're finished, convert the python script you used to populate the database into a PDF and submit it below.

Import Libraries

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random

import datetime
from datetime import datetime, timedelta
import scipy.stats

#import sqlite3
import sqlite3 as sq3

import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
#sets the default autosave frequency in seconds
%autosave 60
sns.set_style('dark')
sns.set(font_scale=1.2)

plt.rc('axes', titlesize=9)
plt.rc('axes', labelsize=14)
plt.rc('xtick', labelsize=12)
plt.rc('ytick', labelsize=12)

pd.set_option('display.max_columns',None)
#pd.set_option('display.max_rows',None)
pd.set_option('display.width', 1000)
pd.option_context('float_format','{:.2f}'.format)

random.seed(0)
np.random.seed(0)
np.set_printoptions(suppress=True)
```

```
Autosaving every 60 seconds
```

In [2]:
```python
%%capture
%load_ext sql
#%sql sqlite://chinook.db
#%sql mysql://studentuser:studentpw@localhost/dognitiondb
#%sql USE dognitiondb
```

Load CSV files

In [3]:
```python
df0 = pd.read_csv("shipping_data_0.csv")
df0
```

Out[3]:

	origin_warehouse	destination_store	product	on_time	product_quantity	driver_identifier
0	d5566b15-b071-4acf-8e8e-c98433083b2d	50d33715-4c77-4dd9-8b9d-ff1ca372a2a2	lotion	True	59	d8da0460-cf39-4f38-9fff-6c9b4e344d8a
1	c42f0de8-b4f0-4167-abd1-ae79e5e18eea	172eb8f3-1033-4fb6-b66b-d0df09df3161	windows	True	28	293ccaec-6592-4f04-aae5-3e238fe62614
2	b145f396-de9b-42f1-9cc9-f5b52c3a941c	65e4544d-42ae-4751-9580-bdcb90e5fcda	skis	True	63	80988f09-91a3-4e1b-8e69-13551c53f318
3	f4372224-759f-43b3-bc83-ca6106bba1af	745bee4e-710c-4538-8df1-5c146e1092a6	bikes	True	47	5f79b402-655f-4d8e-8ff3-5ef05870e0ad
4	49d0edae-9091-41bb-a08d-ab1c66bd08d5	425b7a1a-b744-4c6b-898e-d424dd8cf18e	candy	False	73	58beb5d3-98f8-4077-a964-1f04f7cb11e5
...	...	...	...	...	...	...
105	d2ee1b75-2218-4753-9487-dcca23d667c6	0a994581-341f-43bf-979d-ece1e58de7ec	paint	True	95	a9784b8d-d222-4cdf-93fb-b3886c8033c5
106	6a6d3fce-c5aa-4154-a6a3-b56cb41f709f	403bf915-a897-4918-933b-3996e144e960	snakes	False	54	2fd9a976-bac5-4803-be43-bf93cc618ad1
107	b19cec0d-357e-4c6b-9257-8be52b1c71b5	d3b17672-60fb-443f-a047-2c379132dcb1	alternators	False	20	45c9bd5b-caf6-4ec1-b1eb-09fe615fbdc6
108	d2a2460e-00d1-41f2-84cc-eba01eb88d75	b9f78d5b-79ae-441e-9dbf-592767af34a5	pencil sharpeners	False	7	d7432792-20ad-4a7f-a395-81f04fee89fe
109	75891066-59b4-437b-951f-ec553fb26b94	28fff0d2-38ea-40a7-b2ef-c2a2f7e69370	apples	False	35	cebc86e8-c327-46f7-96b3-35684d169455

110 rows × 6 columns

In [4]:
```python
df0.columns
```

Out[4]:
```
Index(['origin_warehouse', 'destination_store', 'product', 'on_time', 'product_quantity', 'driver_identifier'],
dtype='object')
```

In [5]:
```python
df0.drop(['origin_warehouse', 'destination_store','on_time', 'product_quantity'], axis=1, inplace=True)
```

In [6]:
```python
df0.head()
```

Out[6]:

	product	driver_identifier
0	lotion	d8da0460-cf39-4f38-9fff-6c9b4e344d8a
1	windows	293ccaec-6592-4f04-aae5-3e238fe62614
2	skis	80988f09-91a3-4e1b-8e69-13551c53f318
3	bikes	5f79b402-655f-4d8e-8ff3-5ef05870e0ad
4	candy	58beb5d3-98f8-4077-a964-1f04f7cb11e5

In [7]:
```python
df0.columns = ["name", "id"]
```

In [8]:
```python
df0.head()
```

Out[8]:

	name	id
0	lotion	d8da0460-cf39-4f38-9fff-6c9b4e344d8a
1	windows	293ccaec-6592-4f04-aae5-3e238fe62614
2	skis	80988f09-91a3-4e1b-8e69-13551c53f318
3	bikes	5f79b402-655f-4d8e-8ff3-5ef05870e0ad
4	candy	58beb5d3-98f8-4077-a964-1f04f7cb11e5

In [9]:
```python
df0_new = df0[["id","name"]]
```

In [10]:
```python
df0_new
```

Out[10]:

	id	name
0	d8da0460-cf39-4f38-9fff-6c9b4e344d8a	lotion
1	293ccaec-6592-4f04-aae5-3e238fe62614	windows
2	80988f09-91a3-4e1b-8e69-13551c53f318	skis
3	5f79b402-655f-4d8e-8ff3-5ef05870e0ad	bikes
4	58beb5d3-98f8-4077-a964-1f04f7cb11e5	candy
...	...	...
105	a9784b8d-d222-4cdf-93fb-b3886c8033c5	paint
106	2fd9a976-bac5-4803-be43-bf93cc618ad1	snakes
107	45c9bd5b-caf6-4ec1-b1eb-09fe615fbdc6	alternators
108	d7432792-20ad-4a7f-a395-81f04fee89fe	pencil sharpeners
109	cebc86e8-c327-46f7-96b3-35684d169455	apples

110 rows × 2 columns

In [11]:
```python
#df0_new.to_csv("product.csv",index=False)
```

In [12]:
```python
df1 = pd.read_csv("shipping_data_1.csv")
df1
```

Out[12]:

	shipment_identifier	product	on_time
0	449263b4-6c93-4f19-8b6a-0d99a29fc637	pants	False
1	449263b4-6c93-4f19-8b6a-0d99a29fc637	pants	False
2	449263b4-6c93-4f19-8b6a-0d99a29fc637	pants	False
3	449263b4-6c93-4f19-8b6a-0d99a29fc637	keyboards	False
4	449263b4-6c93-4f19-8b6a-0d99a29fc637	keyboards	False
...	...	...	...
105	c2237ca1-b7e3-40ab-b798-e1ea469301dc	keyboards	True
106	cfa8a834-54bd-4f47-99ca-8912df32913b	animal masks	False
107	cfa8a834-54bd-4f47-99ca-8912df32913b	furniture	False
108	cfa8a834-54bd-4f47-99ca-8912df32913b	furniture	False
109	cfa8a834-54bd-4f47-99ca-8912df32913b	furniture	False

110 rows × 3 columns

In [13]:
```python
df2 = pd.read_csv("shipping_data_2.csv")
df2
```

Out[13]:

	shipment_identifier	origin_warehouse	destination_store	driver_identifier
0	449263b4-6c93-4f19-8b6a-0d99a29fc637	bb75bf7d-c008-4267-bf92-6089cff5fe56	5e9405de-a078-4b00-99c6-96564568b63c	c12025e6-6f9c-4728-8c3c-9f840bde6f1a
1	76e5b84a-9d09-4efb-8b43-a0c932b958bb	372fd2b1-b2a7-4553-b6d7-426a1bc88e56	e34973c8-9ca9-4a06-b497-7a8b49625fc2	85b8d394-a67c-48b6-b1de-55be323ba622
2	b541a47d-89b1-4805-97d0-1988832321f1	469d957f-28ef-4eac-956a-d2a42b06d3ab	fcadc756-61e9-41bb-871b-d3546c5aa981	47bdfc40-f3db-4678-b6a7-43f1e1c2fd32
3	3fc6b63d-27b4-408c-b3b3-e8e94a45b079	cd140190-a53b-4660-a5b4-cc844a6506f0	89ba200c-ca90-443a-b64f-397bce091eae	5ae3e541-2098-45b6-8d94-35d176185606
4	491ee4e8-be80-4f52-802b-d8fe1a6bd487	c6addf8b-eea6-43b8-9040-b5620b1a0d99	7aebe820-8478-4a29-a606-7c59af677e24	1f228b52-7165-4d7f-a731-3f7707aefb2a
5	22768e96-0dad-40d2-8204-3921263c3826	5d64f731-cb01-4992-a27c-a6e1342f4913	d57d76d8-7dca-4ee4-84c0-1745fb4c8779	9136e027-dc50-48b0-b2cf-d01f0c412815
6	f20bbd93-1312-4f70-b257-654056412ec5	abc09fec-2fa0-48f6-b7c4-913620785520	52479603-9957-4e4b-91eb-337c358d1755	85f31f19-81ff-4b03-b862-f2ba16605434
7	192cc6dc-4799-4247-a228-6d198675c00b	bb53f18b-e3c5-48c1-900e-e8ed623ca467	1add84b7-14f5-4857-903b-578408246946	ab78787d-6f8d-48ec-8cef-a7a50ae9c7db
8	e31e22c1-5395-43d8-8a0a-79396d627f66	ee67c3b0-aa89-4b3b-8bbc-9d70695c132b	fa0ce0bb-b0d8-469d-8d42-e1153fc48272	4159e22a-d107-42e6-ba56-f9b65ad8df08
9	6060c04b-921b-4bf9-b2d3-40e57257e5c6	f04f3daf-8ede-4787-a3ad-6ff06274229d	130208de-fef4-46cd-8b9b-1ea5b939895b	d194a942-695a-4e09-9701-490ea8d627f6
10	d2306016-fe82-41f8-a8e1-b06812007036	48c4ca28-4db8-420e-af57-241818a81194	dc042557-cbee-4743-9b72-2c0a34a99cc2	469c402e-d073-49a4-9598-c44a32724643
11	3433af6d-4857-4dfa-88d6-5dcd2a2fce6b	57001f3e-d6be-4031-9295-a6b208c0ad46	49714439-c858-4e61-b7c6-f4d4c848c46b	0966463d-fbac-41cb-b780-0211b98c9beb
12	2fcf115d-068f-4a65-b443-2c5c6fe33fe0	c42221be-4851-42df-9184-1b4f96962bb7	134fb73e-fc99-41e4-92c4-6ecbada5574f	474067dd-c2fd-4bf0-979e-4ecc67cdeafd
13	a39dad1a-4b34-4f50-879e-08bec70d2b36	20efa3c2-c498-4908-8af4-f81c76781912	0167b0c3-60fd-4fbd-b378-fba0ceac39ff	45a23a59-09cb-4906-8293-6014b72de249
14	4fc4c56e-9e44-432a-9d1c-4219271a7844	f156fb67-e7e7-448c-b9c3-0b5f0425d134	2ba952cd-d5bf-4bb4-96fe-02fd4e166d12	a781da7a-1f22-4d1d-8975-37165a13e6ca
15	c9cf4b47-8a85-488d-b33f-d1f921552ea0	c44107fb-70a4-4ff2-bedb-48b9ecfd3d6d	479bfe4c-7137-44dd-bb42-a4ab7900aa24	f851975d-1482-459b-b114-5f73e87a9627
16	2b0bddcd-d73c-4ba6-b26e-aaa785d50be7	026ffd7c-cff2-4daa-ae76-6768d3283861	d33a7b5d-15e5-4ba6-93ef-136bbcbf4946	6e905a32-46a8-4555-8434-a6b16580e873
17	36117159-bb83-4b02-b1d0-07f68fe03be6	950db98c-fb0f-4b78-bbee-bf9ae97faeca	5b9e2a68-7967-46e0-8e35-bc993a1f07e6	b18c8c6e-0aa2-42ce-96d4-0e95f3f04ae5
18	c2237ca1-b7e3-40ab-b798-e1ea469301dc	b377e5d5-563f-475d-8c6d-b9f85ad861fd	fa60bc82-665e-4fe0-8f1f-b8a7675c2e2a	eabecfd5-5ec5-4639-b32d-e204b34af195
19	cfa8a834-54bd-4f47-99ca-8912df32913b	5158fc84-71e0-47a1-84e9-b3e446a391ae	76f02f30-28cd-4f15-88be-9c64860d1fce	aa4c2cd3-0f2c-4982-abd3-be7b06facc87

In [14]:
```python
df3 = pd.merge(left=df2, right=df1, on="shipment_identifier", how="inner")
```

In [15]:
```python
df3.head()
```

Out[15]:

	shipment_identifier	origin_warehouse	destination_store	driver_identifier	product	on_time
0	449263b4-6c93-4f19-8b6a-0d99a29fc637	bb75bf7d-c008-4267-bf92-6089cff5fe56	5e9405de-a078-4b00-99c6-96564568b63c	c12025e6-6f9c-4728-8c3c-9f840bde6f1a	pants	False
1	449263b4-6c93-4f19-8b6a-0d99a29fc637	bb75bf7d-c008-4267-bf92-6089cff5fe56	5e9405de-a078-4b00-99c6-96564568b63c	c12025e6-6f9c-4728-8c3c-9f840bde6f1a	pants	False
2	449263b4-6c93-4f19-8b6a-0d99a29fc637	bb75bf7d-c008-4267-bf92-6089cff5fe56	5e9405de-a078-4b00-99c6-96564568b63c	c12025e6-6f9c-4728-8c3c-9f840bde6f1a	pants	False
3	449263b4-6c93-4f19-8b6a-0d99a29fc637	bb75bf7d-c008-4267-bf92-6089cff5fe56	5e9405de-a078-4b00-99c6-96564568b63c	c12025e6-6f9c-4728-8c3c-9f840bde6f1a	keyboards	False
4	449263b4-6c93-4f19-8b6a-0d99a29fc637	bb75bf7d-c008-4267-bf92-6089cff5fe56	5e9405de-a078-4b00-99c6-96564568b63c	c12025e6-6f9c-4728-8c3c-9f840bde6f1a	keyboards	False

In [16]:
```python
df3.columns
```

Out[16]:
```
Index(['shipment_identifier', 'origin_warehouse', 'destination_store', 'driver_identifier', 'product', 'on_tim
e'], dtype='object')
```

In [17]:
```python
df3.drop(['shipment_identifier','on_time'],axis=1, inplace=True)
```

In [18]:
```python
df3.head()
```

Out[18]:

	origin_warehouse	destination_store	driver_identifier	product
0	bb75bf7d-c008-4267-bf92-6089cff5fe56	5e9405de-a078-4b00-99c6-96564568b63c	c12025e6-6f9c-4728-8c3c-9f840bde6f1a	pants
1	bb75bf7d-c008-4267-bf92-6089cff5fe56	5e9405de-a078-4b00-99c6-96564568b63c	c12025e6-6f9c-4728-8c3c-9f840bde6f1a	pants
2	bb75bf7d-c008-4267-bf92-6089cff5fe56	5e9405de-a078-4b00-99c6-96564568b63c	c12025e6-6f9c-4728-8c3c-9f840bde6f1a	pants
3	bb75bf7d-c008-4267-bf92-6089cff5fe56	5e9405de-a078-4b00-99c6-96564568b63c	c12025e6-6f9c-4728-8c3c-9f840bde6f1a	keyboards
4	bb75bf7d-c008-4267-bf92-6089cff5fe56	5e9405de-a078-4b00-99c6-96564568b63c	c12025e6-6f9c-4728-8c3c-9f840bde6f1a	keyboards

In [19]:
```python
df3["quantity"] = np.random.random_integers(1,101, size=len(df3))
```

In [20]:
```python
df3
```

Out[20]:

	origin_warehouse	destination_store	driver_identifier	product	quantity
0	bb75bf7d-c008-4267-bf92-6089cff5fe56	5e9405de-a078-4b00-99c6-96564568b63c	c12025e6-6f9c-4728-8c3c-9f840bde6f1a	pants	45
1	bb75bf7d-c008-4267-bf92-6089cff5fe56	5e9405de-a078-4b00-99c6-96564568b63c	c12025e6-6f9c-4728-8c3c-9f840bde6f1a	pants	48
2	bb75bf7d-c008-4267-bf92-6089cff5fe56	5e9405de-a078-4b00-99c6-96564568b63c	c12025e6-6f9c-4728-8c3c-9f840bde6f1a	pants	65
3	bb75bf7d-c008-4267-bf92-6089cff5fe56	5e9405de-a078-4b00-99c6-96564568b63c	c12025e6-6f9c-4728-8c3c-9f840bde6f1a	keyboards	68
4	bb75bf7d-c008-4267-bf92-6089cff5fe56	5e9405de-a078-4b00-99c6-96564568b63c	c12025e6-6f9c-4728-8c3c-9f840bde6f1a	keyboards	68
...	...	...	...	...	...
105	b377e5d5-563f-475d-8c6d-b9f85ad861fd	fa60bc82-665e-4fe0-8f1f-b8a7675c2e2a	eabecfd5-5ec5-4639-b32d-e204b34af195	keyboards	65
106	5158fc84-71e0-47a1-84e9-b3e446a391ae	76f02f30-28cd-4f15-88be-9c64860d1fce	aa4c2cd3-0f2c-4982-abd3-be7b06facc87	animal masks	96
107	5158fc84-71e0-47a1-84e9-b3e446a391ae	76f02f30-28cd-4f15-88be-9c64860d1fce	aa4c2cd3-0f2c-4982-abd3-be7b06facc87	furniture	70
108	5158fc84-71e0-47a1-84e9-b3e446a391ae	76f02f30-28cd-4f15-88be-9c64860d1fce	aa4c2cd3-0f2c-4982-abd3-be7b06facc87	furniture	95
109	5158fc84-71e0-47a1-84e9-b3e446a391ae	76f02f30-28cd-4f15-88be-9c64860d1fce	aa4c2cd3-0f2c-4982-abd3-be7b06facc87	furniture	1

110 rows × 5 columns

In [21]:
```python
df3.columns = ["origin","destination","id","product_id","quantity"]
```

In [22]:
```python
df3.head(1)
```

Out[22]:

	origin	destination	id	product_id	quantity
0	bb75bf7d-c008-4267-bf92-6089cff5fe56	5e9405de-a078-4b00-99c6-96564568b63c	c12025e6-6f9c-4728-8c3c-9f840bde6f1a	pants	45

In [23]:
```python
df3_new = df3[["id","product_id","quantity","origin","destination"]]
```

In [24]:
```python
df3_new.head(1)
```

Out[24]:

	id	product_id	quantity	origin	destination
0	c12025e6-6f9c-4728-8c3c-9f840bde6f1a	pants	45	bb75bf7d-c008-4267-bf92-6089cff5fe56	5e9405de-a078-4b00-99c6-96564568b63c

In [25]:
```python
#df3_new.to_csv("shipment.csv", index=False)
```

Upload Data to Database

In [27]:
```python
con = sq3.connect("shipment_database.db")
```

In [28]:
```python
con
```

Out[28]:
```
<sqlite3.Connection at 0x1d280607300>
```

In [31]:
```python
df0_new.to_sql("product", con, index=False)  #Load to database
```

In [32]:
```python
df3_new.to_sql("shipment", con, index=False)  #Load to database
```

In [33]:
```python
con.execute("Select * FROM sqlite_master").fetchall()
```

Out[33]:
```
[('table',
  'product',
  'product',
  2,
  'CREATE TABLE "product" (\n"id" TEXT,\n  "name" TEXT\n)'),
 ('table',
  'shipment',
  'shipment',
  4,
  'CREATE TABLE "shipment" (\n"id" TEXT,\n  "product_id" TEXT,\n  "quantity" INTEGER,\n  "origin" TEXT,\n  "des
tination" TEXT\n)')]
```

In [34]:
```python
pd.read_sql_query("Select * FROM sqlite_master", con) #Check all tables inside database
```

Out[34]:

	type	name	tbl_name	rootpage	sql
0	table	product	product	2	CREATE TABLE "product" (\n"id" TEXT,\n "name"...
1	table	shipment	shipment	4	CREATE TABLE "shipment" (\n"id" TEXT,\n "prod...

In [35]:
```python
con.close()
```

Python code done by Dennis Lam